Module 6 – Error Handling

Overview

The main purpose of this lab is to familiarise yourself with key declaring and raising errors, then producing code that catches errors.

- Declare an error message with a defined error number and message
- Throw errors instead of printing an error message
- Catch an error from within code

Objectives

At the end of this lab, you will be able to:

- Declare an error within the sys.messages system table
- Raise an previously declared error
- Throw an error
- Produce code that catches errors and deals with the error efficiently

Setup: Launch SQL Server Management Studio

1. On the Start menu, click All Programs, click Microsoft SQL Server 2014 and then click SQL Server Management Studio.
2. The Microsoft SQL Server Management Studio window opens, and then the Connect to Server dialog box will appear.
3. In the Connect to Server dialog box, click Connect to accept the default settings.
4. On the toolbar, click New Query, and either select the QATSQLPLUS database in the Available Databases box, or type USE QATSQLPLUS in the query window.
5. If you wish, you may save your queries to My Documents or the desktop. All modules are separate and you will not require any queries from this module in any later module.

Exercise 1: Declared error messages

In this exercise, you will register an error message and raise the error.

The main tasks for this exercise are as follows:

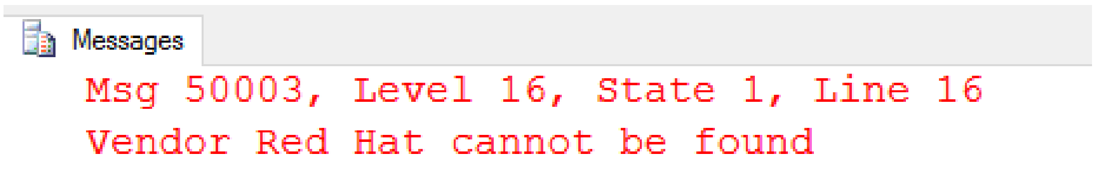
1. Declare the error message.
2. Raise the error message.

Task 1: Create the error message

1. Write a TSQL statement to declare the error message with the following parameters:
   - Error number : 50003
   - Error level : 16
   - Error message : Vendor %s cannot be found
2. Execute the query. The result should be “Command(s) completed successfully”. If an error 15043 is returned then the error number 50003 is already declared. Change the number to another above 50000 and try again.
3. Keep the query window open for the following tasks.

Task 2: Raise the declared error

1. Write a TSQL statement to raise the error message with the following parameters:
   - Error number : 50003 (unless you changed this in the previous task)
   - Error level : 16
   - Error state : 1
   - String : “Red Hat”
2. Test the query. The result should be:



```
Msg 50003, Level 16, State 1, Line 16
Vendor Red Hat cannot be found
```

3. Keep the query window open for the following tasks.

Exercise 2: Throwing Errors

In this exercise, you will alter an existing code to throw errors rather than using print.

The main tasks for this exercise are as follows:

1. Identify the lines of code to be replaced.
2. Change the code to utilise the THROW statement.

Task 1: Analyse the code

1. Analyse the code below to select the line(s) that need replacing by the THROW statement:

```
--TASK 1:
    DECLARE @Vendor VARCHAR(100)
    SET @Vendor = 'QA'

    IF @Vendor IS NULL
        BEGIN
            PRINT 'Vendor must not be NULL'
            RETURN
        END

    IF NOT EXISTS (SELECT * FROM dbo.Vendor WHERE VendorName = @Vendor)
        BEGIN
            PRINT 'Vendor ' + @Vendor + ' does not exist'
            RETURN
        END

    SELECT *
        FROM dbo.Course AS C
            INNER JOIN dbo.Vendor AS V
                ON C.VendorID = V.VendorID
        WHERE VendorName = @Vendor
GO
```

Task 2: Update the code

1. Open the query “s:\qatsqlplus\M06 E02 Raising Errors.sql”.
2. Update the code to THROW rather than PRINT errors.
   - The error number should be 54000 and the state should be 1.
   - If the @Vendor is NULL then the error message should be “Vendor must not be NULL”
   - if the @Vendor does not exist then the error message should be “Vendor cannot be found”
3. Test the code with:
   - @Vendor = ‘QA’ (4 rows)
   - @Vendor = ‘AQ’ (error – Vendor cannot be found)
   - @Vendor = NULL (error – Vendor must not be NULL)
4. Keep the query window open for the following tasks.

Task 3 (if time permits): Update the code again

1. Open the query “s:\qatsqlplus\M06 E02 Raising Errors.sql”.
2. Update the code to enhance the second error message
   - If the @Vendor does not exist then the error message should be “Vendor <insert @Vendor value> cannot be found”
   - Use FORMATMESSAGE to create the message text
3. Test the code with:
   - @Vendor = ‘QA’ (4 rows)
   - **@Vendor = ‘AQ’ (error – Vendor AQ cannot be found)**
   - @Vendor = NULL (error – Vendor must not be NULL)
4. Keep the query window open for the following tasks.

Exercise 3: Throwing Errors

In this exercise, you will catch errors thrown by an update due to table constraints.

The main tasks for this exercise are as follows:

1. Identify possible issues.
2. Change the code to utilise the THROW statement.

Task 1: Analyse the code

1. Analyse the code below to view potential issue(s).
2. The Vendor table used in this exercise is defined as:
    - dbo.Vendor
        - VendorID primary key int not null
        - VendorName varchar(100) not null
        - ContactName varchar(100) not null
        - PhoneNumber varchar(15) nulls allowed
3. The original code:

```sql
DECLARE @Vendor VARCHAR(30) = 'QA'
UPDATE dbo.Vendor
    SET VendorName = @Vendor
    WHERE VendorID = 1
```

4. What issues may arise from the code given the table designs?

Task 2: Update the code

1. Open the query “s:\qatsqlplus\M06 E03 Try Catch.sql”.
2. Alter the code to catch errors from within the code and return a formatted message with error number = 60000 and state = 1.
3. Execute the code with different parameters:
   - @Vendor = ‘QA’
     - 1 row(s) updated
   - @Vendor = NULL
     - Error message with error number 60000

Answers

The answers below are for example only. Coding style and order of columns should not matter.

Ex 1 Task 1

```sql
--TASK 1:
EXEC sp_addmessage 50003,16,'Vendor %s cannot be found'
GO
```

Ex 1 Task 2

```sql
--TASK 2:
RAISERROR(50003,16,1,'Red Hat')
GO
```

Ex 2 Task 1

```sql
--TASK 1:
DECLARE @Vendor VARCHAR(100)
SET @Vendor = 'QA'

IF @Vendor IS NULL
  BEGIN
   PRINT 'Vendor must not be NULL'
   RETURN
  END

IF NOT EXISTS (SELECT * FROM dbo.Vendor WHERE VendorName = @Vendor)
  BEGIN
   PRINT 'Vendor ' + @Vendor + ' does not exist'
   RETURN
  END

SELECT *
  FROM dbo.Course AS C
   INNER JOIN dbo.Vendor AS V
    ON C.VendorID = V.VendorID
  WHERE VendorName = @Vendor
GO
```

Ex 2 Task 2

```sql
--TASK 2:
DECLARE @Vendor VARCHAR(100)
SET @Vendor = 'QA'

IF @Vendor IS NULL
  BEGIN;
    THROW 54000,'Vendor must not be NULL',1
    RETURN
  END

IF NOT EXISTS (SELECT * FROM dbo.Vendor WHERE VendorName = @Vendor)
  BEGIN;
    THROW 54000,'Vendor cannot be found',1
    RETURN
  END

SELECT *
  FROM dbo.Course AS C
    INNER JOIN dbo.Vendor AS V
      ON C.VendorID = V.VendorID
  WHERE VendorName = @Vendor
GO
```

Ex 2 Task 3	
	```
	--TASK 3:
	DECLARE @Vendor VARCHAR(100)
	SET @Vendor = 'AQ'

	IF @Vendor IS NULL
	  BEGIN;
	    THROW 54000, 'Vendor must not be NULL',1
	    RETURN
	  END

	IF NOT EXISTS (SELECT * FROM dbo.Vendor WHERE VendorName = @Vendor)
	  BEGIN
	    DECLARE @msg VARCHAR(100)
	    SET @msg = FORMATMESSAGE('Vendor %s cannot be found.',@Vendor);
	    THROW 54000,@msg ,1
	    RETURN
	  END

	SELECT *
	  FROM dbo.Course AS C
	    INNER JOIN dbo.Vendor AS V
	      ON C.VendorID = V.VendorID
	  WHERE VendorName = @Vendor
	```

Ex 3 Task 1	• The VendorName cannot be NULL so a NULL parameter would cause an issue.

Ex 3 Task 2	`BEGIN TRY`
	`  UPDATE dbo.Vendor`
	`    SET VendorName = NULL`
	`    WHERE VendorID = 1`
	`END TRY`
	`BEGIN CATCH`
	`  THROW 60000,'Error occurred within the procedure',1`
	`END CATCH`
	`GO`